WHAT IS CLAIMED IS:

1. A method for converting strongly typed objects for use in weakly typed frameworks, the method comprising:

receiving an object input parameter that is a strongly typed object for conversion;

receiving a keyword input parameter for identifying properties and fields of the strongly typed object to be used in conversion;

invoking a mapping method for determining a Property Definition Dictionary and a Field Definition Dictionary based upon the object input parameter and the keyword input parameter;

creating the weakly typed object;

modifying property values associated with the weakly typed object based upon name and property definitions in the Property Definition Dictionary; and

modifying field values associated with the weakly typed object based upon name and field definitions in the Field Definition Dictionary.

2. A method as recited in Claim 1, wherein an object input parameter identifies the weakly typed object resulting from conversion.

3. A method as recited in Claim 1, wherein an input to the mapping method is a write control mode parameter that is set to false.

4. A method as recited in Claim 3, wherein a parameter output by the mapping method is a type output.

5. A method for converting objects representing structured data input comprising the steps of:

examining, during runtime by reflection, attributes associated with an object for runtime information about code symbols, wherein the attributes have tags that add descriptions to the code symbols; and

converting the object associated with the code symbols based upon a result of the reflection, wherein the attributes include at least one attribute having a plurality of attribute parameters that specify how a conversion on the object is to be done.

6. A method as recited in Claim 5, wherein the plurality of attribute parameters are an auto parameter, a name parameter and an access parameter that assigned to an environment entity.

7. A method as recited in Claim 6, further comprising the step of exposing only public properties of an associated class to conversion if the auto parameter is true.

8. A method as recited in Claim 6, wherein the name parameter specifies a specific name for the object being converted.

9. A method as recited in Claim 6, wherein the name parameter is associated with a class of the converted object such that the name parameter has a value indicating that a name of the object after conversion is derived from a name of the object prior to conversion.

10. A method as recited in Claim 6, wherein the access parameter is associated with at least one of a group consisting of a class, a property and a field.

11. A method as recited in Claim 10, further comprising ignoring the access parameter when applied to a class.

12. A method as recited in Claim 6, further comprising giving assigned values to the access parameter when the access parameter applies to a property or field, wherein the assigned values are selected from the group consisting of both, read, and write.

13. A method as recited in Claim 12, wherein the both value indicates that the property or field is used for read and write operations, the read value indicates the property or field is used only for read operations, and the write value indicates the property or field is used only for write operations for allowing one property or field to be used for conversion during a read operation and another property or field to be used for conversion during a write operation.

14. A method as recited in Claim 6, wherein the access parameter provides privileged access to a private field implementation for write access during a conversion, thereby maintaining encapsulation for the remainder of environment code and allowing special access only to a conversion process.

15. A mapping method for ascertaining attributes related to objects in order to optimize utilization of a programming environment on an associated platform, the mapping method comprising the steps of:

receiving an object input parameter that identifies a strongly typed object involved in a conversion process;

receiving an iswritemode input parameter that identifies a type of conversion;

setting an astype output parameter based upon an attribute associated with a target object, wherein the astype output parameter indicates a name of a weakly typed object;

using reflection on the strongly typed object to obtain a first list representing properties of the object input parameter;

filtering the first list to include only readable and writable properties to create a first subset of the first list;

assigning and storing default names based upon the first subset in a dictionary cache along with a class name and the iswritemode input parameter,

using reflection on the strongly typed object to obtain a second list of fieldinfo objects associated with the object input parameter;

filtering the second list to create a second subset of the second list based upon criteria;

if the iswritemode input parameter is true, then filtering the second subset to create a third list that includes only fieldinfo objects that represent fields that have an associated ActionScript attribute with an Access property set to Write or Both;

if the iswritemode input parameter is false, then filtering the second subset to create a third list that includes only those fieldinfo objects that represent fields that have an associated ActionScript attribute with the Access property set to Read or Both;

assigning names for each FieldInfo object in the third list;

storing the names for each FieldInfo object along with its associated FieldInfo object in a dictionary cache; and

using the dictionary cache to complete attributes associated with the target object.

16. A mapping method as recited in Claim **15**, wherein the criteria for the second subset list is only those fieldinfo objects that represent fields that have an associated ActionScript attribute.

17. A mapping method as recited in Claim **15**, wherein the dictionary cache includes:

a property definition dictionary having a property key and a property value, wherein the property key is the name of the weakly typed object and the property value is a PropertyInfo object furnished by reflection that is associated with that name; and

a field definition dictionary having a field key and a field value, wherein the field key is a name of the field's representation in the weakly typed object and the field value is the FieldInfo object furnished by reflection associated with that name.

18. A mapping method as recited in Claim **15**, wherein a true value for the iswritemode input parameter indicates that output parameters are for converting a weakly typed object to the target object of a strongly typed object and a false value indicates that output parameters are for converting a strongly typed object to the target object of a weakly typed object.

19. A mapping method as recited in Claim **15**, wherein the step of setting an astype output parameter is based upon retrieving a class name of the target object through reflection on the strongly typed object.

20. A mapping method as recited in Claim **15**, further comprising the step of storing and using a Keyword input parameter to identify the dictionary cache in subsequent calls to the mapping method.

21. A mapping method as recited in Claim **20**, wherein if the Keyword input parameter is specified, the second list is further filtered to include only those FieldInfo objects that represent fields that have an associated ActionScript attribute with a Keyword property set to a value of the Keyword input parameter.

22. A method of responding to communications received in a remoting environment by a system operating in a web services environment, the method comprising:

(a) receiving, from a communications appliance, a communication including a first program object compatible with the remoting environment, said first program object comprising at least one attribute; and

(b) converting said first program object into a second program object compatible with the web services environment, by a process determined in response to said at least one attribute.

23. A method of providing multimedia content at a communications appliance, the method comprising:

(a) executing at said communications appliance an S-module operable to send to said remote system a program object comprising at least one attribute for controlling a conversion of said program object at said remote system to permit said remote system to handle said program object; and

(b) causing said communications appliance to receive a multimedia object determined by said program object, from said remote system; and

(c) causing said communications appliance to present multimedia content defined by said multimedia object at said communications appliance.

24. A server for facilitating providing information in response to a user having a client associated therewith, wherein the server and the client communicate via a distributed computing network, and wherein the server comprises:

(a) a memory storing an instruction set and a plurality of s-modules; and

(b) a processor for running the instruction set, the processor being in communication with the memory and the distributed computing network, wherein the processor is operative to:

(i) receive a content parameter from the client;

(ii) create and open a window on the client;

(iii) designate an s-module based upon a content parameter;

(iv) send the designated s-module to the client;

(v) receive a reply based upon receipt of the designated s-module; and

(vi) cause content associated with the designated s-module to be displayed in the window.

25. A method for converting values associated with a weakly typed object into properties associated with a strongly typed object in accordance with a strongly typed framework, the method comprising:

creating the strongly typed object;

determining the properties according to the attribute parameters associated with the strongly typed object by reflection; and

populating the properties based upon the values associated with the weakly typed object.

26. A method as recited in Claim 25, wherein at least one of the attribute parameters attached to the strongly typed object is a Name parameter that identifies the weakly typed object.

27. A method as recited in Claim 25,

wherein at least one of the values associated with the weakly typed object is a type property, and

the weakly typed object matches the strongly typed object if the type property of the weakly typed object matches the Name parameter.

28. A method as recited in Claim 27,

wherein at least one of the attribute parameters attached to the strongly typed object is a name of a class that defines the strongly typed object, and

the weakly typed object matches the strongly typed object if the type property of the weakly typed object matches the name of the class that defines the strongly typed object.

29. A method as recited in Claim 25, wherein the determining the properties step returns mapping information in the form of a Property Definition Dictionary, Field Definition Dictionary and a type output parameter.

30. A method as recited in Claim **29**, if a type property of the weakly typed object is identical to the type output parameter of the mapping information, then matching an object entry name associated with the strongly typed object according to name and property definitions in the Property Definition Dictionary.

31. A server for converting a strongly typed object for use in a weakly typed environment, wherein the server is operatively connected to a distributed computing network, and wherein the server comprises:

(a) a memory storing an instruction set; and

(b) a processor for running the instruction set, the processor being in communication with the memory and the distributed computing network, wherein the processor is operative to:

i) receiving an object input parameter that is a strongly typed object for conversion;

ii) receiving a keyword input parameter for identifying properties and fields of the strongly typed object to be used in conversion;

iii) invoking a mapping method for determining a Property Definition Dictionary and a Field Definition Dictionary based upon the object input parameter and the keyword input parameter;

iv) creating the weakly typed object;

v) modifying property values associated with the weakly typed object based upon name and property definitions in the Property Definition Dictionary; and

vi) modifying field values associated with the weakly typed object based upon name and field definitions in the Field Definition Dictionary.

32. A system for ascertaining attributes related to objects in order to optimize utilization of a programming environment on an associated platform, the system comprising:

first means for receiving an object input parameter that identifies a strongly typed object involved in a conversion process;

second means for receiving an iswritemode input parameter that identifies a type of conversion;

third means for setting an astype output parameter based upon an attribute associated with a target object, wherein the astype output parameter indicates a name of a weakly typed object;

fourth means for using reflection on the strongly typed object to obtain a first list representing properties of the object input parameter;

fifth means for filtering the first list to include only readable and writable properties to create a first subset of the first list;

sixth means for assigning and storing default names based upon the first subset in a dictionary cache along with a class name and the iswritemode input parameter,

seventh means for using reflection on the strongly typed object to obtain a second list of fieldinfo objects associated with the object input parameter;

eigth means for filtering the second list to create a second subset of the second list based upon criteria, wherein

if the iswritemode input parameter is true, then filtering the second subset to create a third list that includes only fieldinfo objects that represent fields that have an associated ActionScript attribute with an Access property set to Write or Both;

if the iswritemode input parameter is false, then filtering the second subset to create a third list that includes only those fieldinfo objects that represent fields that have an associated ActionScript attribute with the Access property set to Read or Both;

ninth means for assigning names for each FieldInfo object in the third list;

tenth means for storing the names for each FieldInfo object along with its associated FieldInfo object in a dictionary cache; and

eleventh means for using the dictionary cache to complete attributes associated with the target object.

33. A system as recited in Claim **32**, wherein the first, second, third, fourth, fifth, sixth, seventh, eigth, ninth, tenth and eleventh means are a computer.

34. A program product for use in a computer system that executes program steps recorded in a computer-readable medium to convert values associated with a weakly typed object into properties associated with a strongly typed object in accordance with a strongly typed framework, the program product comprising:

a recordable media for storing a program; and the program of computer-readable instructions executable by the computer system to perform steps including:

creating the strongly typed object;

determining the properties according to the attribute parameters associated with the strongly typed object by reflection; and

populating the properties based upon the values associated with the weakly typed object.